Solved Questions User Study		
1. What is the difference between Ruby and Python versions of“self”? (159990)	Ground Truth	Python is designed to support more than just object-oriented programming. Preserving the same interface between methods and functions lets the two styles interoperate more cleanly. Ruby was built from the ground up to be object-oriented. Even the literals are objects (evaluate 1.class and you get Fixnum). The language was built such that self is a reserved keyword that returns the current instance wherever you are. If you're inside an instance method of one of your class, self is a reference to said instance. If you're in the definition of the class itself (not in a method), self is the class itself:
	Ours	Python is designed to support more than just object-oriented programming. Preserving the same interface between methods and functions lets the two styles interoperate more cleanly. Ruby was built from the ground up to be object-oriented. Even the literals are objects (evaluate 1.class and you get Fixnum). The language was built such that self is a reserved keyword that returns the current instance wherever you are. If you're inside an instance method of one of your class, self is a reference to said instance. If you're in the definition of the class itself (not in a method), self is the class itself:

2. What is the pythonic way to avoid default parameters that are empty lists?(366422)	Ground Truth	def my_func(working_list=None):
		  if working_list is None:
		  working_list = []
		
		  working_list.append("a")
		  print(working_list)
		
		The docs say you should use None as the default and explicitly test for it in the body of the function.

	Ours	def my_func(working_list=None):
		  if working_list is None:
		  working_list = []
		
		  working_list.append("a")
		  print(working_list)
		
		The docs say you should use None as the default and explicitly test for it in the body of the function.

3.How do I get the time a file was last modified in Python? (375154)	Ground Truth	os.stat() import os filename = "/etc/fstab" statbuf = os.stat(filename) print("Modification time: {}".format(statbuf.st_mtime)) Linux does not record the creation time of a file (for most fileystems).
	Ours	os.path.getmtime(filepath) or os.stat(filepath).st_mtime
4. Prototype based object orientation. The good, the bad and the ugly? (385403)	Ground Truth	Prototype-based OO lends itself poorly to static type checking, which some might consider a bad or ugly thing. Prototype-based OO does have a standard way of creating new objects, you clone and modify existing objects. You can also build factories, etc. I think what people like most (the "good") is that prototype-based OO is very lightweight and flexible, offering a very high power-to-weight ratio. For tips on how to use prototype-based OO, a great place to start is the original Self paper on The Power of Simplicity.
	Ours	Prototype-based OO lends itself poorly to static type checking, which some might consider a bad or ugly thing. Prototype-based OO does have a standard way of creating new objects, you clone and modify existing objects. You can also build factories, etc. I think what people like most (the "good") is that prototype-based OO is very lightweight and flexible, offering a very high power-to-weight ratio. For tips on how to use prototype-based OO, a great place to start is the original Self paper on The Power of Simplicity.

5. How do you debug Mako templates? (390409)	Ground Truth	Mako actually provides a VERY nice way to track down errors in a template:
		
		from mako import exceptions
		try:
		  template = lookup.get_template(uri)
		  print template.render()
		except:
		  print exceptions.html_error_template().render()

	Ours	Mako actually provides a VERY nice way to track down errors in a template:
		
		from mako import exceptions
		try:
		  template = lookup.get_template(uri)
		  print template.render()
		except:
		  print exceptions.html_error_template().render()

6. Is there a pure Python Lucene? (438315)	Ground Truth	Whoosh is a new project which is similar to lucene, but is pure python.
	Ours	Yes, Twisted includes a framework for building SMTP servers. There's a simple Twisted-based email server available here (also see here for some information about its development). If you want something closer to a mail application server, there's Lamson.
7. How to automatically reload a python file when it is changed (780526)	Ground Truth	Take at look at CherryPy's Autoreload feature. I think it looks quite simple and always worked well for me.
	Ours	You can use a shutdown hook to execute the other script once the first is completed with atexit atexit.register(lambda: execfile('other.py')) # pass function to execute other file
8. How do I exchange keys with values in a dictionary? (1031851)	Ground Truth	Python 2: res = dict((v,k) for k,v in a.iteritems()) Python 3 (thanks to @erik): res = dict((v,k) for k,v in a.items())
	Ours	Well, once you have a key i, the value for that key is aDictionary[i], so just extend your testing. There are more optimal ways of doing this, that don't do the look-up as explicitly, but the above is straight-forward and easy to understand, and should be fine for many cases.

Solved Questions User Study		
9. Python unittest: how to run only part of a test file? (1068246)	Ground Truth	The default unittest.main() uses the default test loader to make a TestSuite out of the module in which main is running. You don't have to use this default behavior. You can, for example, make three unittest.TestSuite instances. 1.The "fast" subset. fast = TestSuite() fast.addTests(TestFastThis) fast.addTests(TestFastThat) 2. The "slow" subset. slow = TestSuite() slow.addTests(TestSlowAnother) slow.addTests(TestSlowSomeMore) 3. The "whole" set. alltests = unittest.TestSuite([fast, slow]) Note that I've adjusted the TestCase names to indicate Fast vs. Slow. You can subclass unittest.TestLoader to parse the names of classes and create multiple loaders.Then your main program can parse command-line arguments with optparse or argparse (available since 2.7 or 3.2) to pick which suite you want to run, fast, slow or all.
	Ours	The default unittest.main() uses the default test loader to make a TestSuite out of the module in which main is running. You don't have to use this default behavior. You can, for example, make three unittest.TestSuite instances. 1.The "fast" subset. fast = TestSuite() fast.addTests(TestFastThis) fast.addTests(TestFastThat) 2. The "slow" subset. slow = TestSuite() slow.addTests(TestSlowAnother) slow.addTests(TestSlowSomeMore) 3. The "whole" set. alltests = unittest.TestSuite([fast, slow]) Note that I've adjusted the TestCase names to indicate Fast vs. Slow. You can subclass unittest.TestLoader to parse the names of classes and create multiple loaders.Then your main program can parse command-line arguments with optparse or argparse (available since 2.7 or 3.2) to pick which suite you want to run, fast, slow or all.
10. Do I test a class that does nothing? (1127626)	Ground Truth	Of course you can test a class that doesn't do anything. You test that it does, in fact, not do anything.In practice, that means: 1. it exists and can be instantiated; 2. it doesn't throw an exception when used; call every method and simply assert that they succeed. This also doubles as checking that the class does, in fact, define every method that it's expected to. Don't run this test on DummyLog; use it as a generic test and run it on all loggers. When you add another method to the real Log class a year from now and forget to add it to DummyLog, the test will notice. (As long as you do remember to add a general test for the method, of course, but hopefully adding the test should be habitual, even if you forget about related classes.)
	Ours	White box tests can be useful to detect some regression or to assert that a specific action is made. For example you can verify that you don't interact with your DB in this particular case or that you've correctly called the notification service. However, the drawback is that you're likely to change the test when you change the code, because you're test is very tied to the implementation. This can be painful when you are refactoring, because you also need to refactor the test. You could forget an assertion or a step and create a false positive test with a regression. I would use it only if it makes sens and if you need it to assert what's going on in details. You can search on the web TDD: London vs Detroit. You'll find interesting stuff.
11. In Python, how do I convert all of the items in a list to floats? (1614236)	Ground Truth	[float(i) for i in lst] to be precise, it creates a new list with float values. Unlike the map approach it will work in py3k.
	Ours	If you want to keep the integers as they are in the list while just changing the strings to unicode, you can do x = ['Some strings.', 1, 2, 3, 'More strings!'] y = [i.decode('UTF-8') if isinstance(i, basestring) else i for i in x] which gets you [u'Some strings.', 1, 2, 3, u'More strings!']
12. Python: How do I read and parse a unicode utf-8 text file?(1862963)	Ground Truth	That is a BOM EDIT, from the coments, it seems to be a utf-16 bom codecs.open('foo.txt', 'r', 'utf-16') should work.

	Ours	It looks as if your input file is UTF-8 encoded so specify UTF-8 encoding when you open the file (Python3 is assumed as per your reference):
		with open("input.txt", "r", encoding='utf8') as f:
		  text = f.read()
		text will contain the content of the file as a str (i.e. unicode string). Now you can write it in unicode escaped form directly to a file by specifying encoding='unicode-escape':
		with open('output.txt', 'w', encoding='unicode-escape') as f:
		  f.write(text)

13. How can I create a regular expression in Python? (1895026)	Ground Truth	Try something like this: r'[a-zA-Z0-9]+_[^_]+_[a-zA-Z0-9]+\.[a-zA-Z0-9]+'
	Ours	https://www.debuggex.com is also pretty good. It's an online Python (and a couple more languages) debugger, which has a pretty neat visualization of what does and what doesn't match. A pretty good resource if you need to draft a regexp quickly.

14. Python: For each list element apply a function across the list ?493367)	Ground Truth	You can do this using list comprehensions and min() (Python 3.0 code):
		>>> nums = [1,2,3,4,5]
		>>> [(x,y) for x in nums for y in nums]
		>>> min(_, key=lambda pair: pair[0]/pair[1])
		
		Note that to run this on Python 2.5 you'll need to either make one of the arguments a float, or do from __future__ import division so that 1/5 correctly equals 0.2 instead of 0.

	Ours	Using list comprehension:
		>>> oldList = [1, 2, 3, 4]
		>>> newList = [x for x in oldList for _ in range(2)]
		>>> newList
		Above list comprehension is similar to following nested for loop.
		newList = []
		for x in oldList:
		  for _ in range(2):
		  newList.append(x)

15. Loading SQL dump before running Django tests	Ground Truth	You may need to look into defining a custom test runner. There's some info here:https://docs.djangoproject.com/en/dev/topics/testing/advanced/#other-testing-frameworks Basically I think you can just copy the default test runner from django.test.simple.run_tests and then modify it to suit your needs. I've not done this before, but from my understanding that would be the way to customize this.

Solved Questions User Study		
(1979692)	Ours	You may need to look into defining a custom test runner. There's some info here:https://docs.djangoproject.com/en/dev/topics/testing/advanced/#other-testing-frameworks Basically I think you can just copy the default test runner from django.test.simple.run_tests and then modify it to suit your needs. I've not done this before, but from my understanding that would be the way to customize this.
16. Python: How do you format a string % number with str.format() ? (2014012)	Ground Truth	It would be like {0:01.2f}, I believe. See http://docs.python.org/library/string.html#format-string-syntax. Edit: added in your zero-padding.

	Ours	If you are using it in a formatted string with the format() method which is preferred over the older style '% formatting
		>>> 'One hundred and twenty three with three leading zeros {0:06}.'.format(123)
		'One hundred and twenty three with three leading zeros 000123.'
		
		Here is an example with variable width,
		>>> '{num:0{width}}'.format(num=123, width=6)
		'000123'
		
		You can even specify the fill char as a variable
		>>> '{num:{fill}{width}}'.format(num=123, fill='0', width=6)
		'000123'

17. What permissions are required for subprocess.Popen? (2066068)	Ground Truth	Assuming that permissions on parent folders are correct (i.e. all parent folders should have +x permission), try adding: shell=True to the Popen command such as: subprocess.Popen(("/Users/jondoe/development/mdb-export", mdb.name, tbl,), stdout=csv, shell=True)
	Ours	Assuming that permissions on parent folders are correct (i.e. all parent folders should have +x permission), try adding: shell=True to the Popen command such as: subprocess.Popen(("/Users/jondoe/development/mdb-export", mdb.name, tbl,), stdout=csv, shell=True)

18.String count with overlapping occurrences (2970520)	Ground Truth	Well, this might be faster since it does the comparing in C:
		def occurrences(string, sub):
		  count = start = 0
		  while True:
		  start = string.find(sub, start) + 1
		  if start > 0:
		  count+=1
		  else:
		  return count

	Ours	I have tried to optimise the code. This code works for overlapping while for non-overlapping matches count can be used.
		string = 'dadada'
		substring = 'dad'
		results = 0
		sub_len = len(substring)
		for i in range(len(string)):
		  if string[i:i+sub_len] == substring:
		  results += 1
		print results
		…

19. Python script reading from a csv file (3107793)	Ground Truth	You're importing the csv module but never use it. Why? If you do import csv reader = csv.reader(open(file, "rb"), dialect="excel") # Python 2.x # Python 3: reader = csv.reader(open(file, newline=""), dialect="excel") you get a reader object that will contain all you need; the first row will contain the headers, and the subsequent rows will contain the data in the corresponding places. Even better might be (if I understand you correctly): import csv reader = csv.DictReader(open(file, "rb"), dialect="excel") # Python 2.x # Python 3: reader = csv.DictReader(open(file, newline=""), dialect="excel")

	Ours	First, make sure you converted the file from Excel to csv, using the Save As menu from Excel. Simply changing the extension doesn't work. The output you are seeing is data from Excel's native format.
		Once you have converted the files, use the csv module:
		import csv
		for filename in os.listdir(INPUT_DIR):
		  with open(os.path.join(INPUT_DIR,filename), dialect='excel-tab') as infile:
		  reader = csv.reader(infile)
		  for row in reader:
		  print row
		If you want to read raw Excel files, use the xlrd module. Here is a sample that shows how to read Excel files.

20.How to detect the OS default language in python? (3425294)	Ground Truth	You could use the getdefaultlocale function in the locale module. It returns the language code and encoding of the system default locale in a tuple:
		>>> import locale
		>>> locale.getdefaultlocale()
		('en_GB', 'cp1252')

	Ours	Consider checking the PROCESSOR_ARCHITECTURE key in os.environ, and seeing if it's set to x86 or not. Alternatively, you can use the platorm.architecture() call, but point it at a different executable from the Python one (since you're required to run under 32bit Python).
21. How to remove lines from stdout in python? (3732928)	Ground Truth	readlines provides all the data allLines = [line for line in stdout.readlines()] data_no_firstfour = "\n".join(allLines[4:])
	Ours	Your result variable is a list. I think you want to join the results into a single string and print that. You can do that with str.join() like this: print ''.join(result) This will result in the following output.
22. How to update a file in python egg (3736043)	Ground Truth	Just build a new egg (with python setup.py bdist_egg), reinstall the egg and restart your trac instance.
	Ours	Just build a new egg (with python setup.py bdist_egg), reinstall the egg and restart your trac instance.

Solved Questions User Study		
23. Why does Python skip elements when I modify a list while iterating over it?? (742371)	Ground Truth	This is a well-documented behaviour in Python, that you aren't supposed to modify the list being iterated through. Try this instead: for i in x[:]: x.remove(i) The [:] returns a "slice" of x, which happens to contain all its elements, and is thus effectively a copy of x.
	Ours	You edit the list while loop over it. Since you only keep pointers for looping you remeber positions, but they change since elements vanish. Try remebering the elements you want to delete by something like: … Using List Comprehensions from the start leads to: dt_list = [3600, 2700, 1800, 900] dt_list = [x for x in dt_list if (3600/x).is_integer()]

24. ImportError: No module named *** in python (3992952)**	Ground Truth	These files are not on sys.path. It should have been though.
		If you want to access them from the interpreter, you will need to add the location to sys.path
		>>> import sys
		>>> print sys.path
		>>> sys.path.append('C:\\Users\\Cube\\Documents\\Python')
		>>> import reduc
		You could also include the path in environment variable - PYTHONPATH
		See the details on module search path here : … Also look at (PYTHONPATH) environment variable details here:

	Ours	When dealing with version ambiguity, remember that pip is a python module. Once you're confident that python is the python installation that your IDE is running, run python --version python -m pip install seaborn >pip3 may be pointing to an old or different python installation.

25. How do we determine the number of days for a given month in python(4938429)	Ground Truth	Use calendar.monthrange:
		>>> from calendar import monthrange
		>>> monthrange(2011, 2)
		(1, 28)
		Just to be clear, monthrange supports leap years as well:
		>>> from calendar import monthrange
		>>> monthrange(2012, 2)
		(2, 29)
		As @mikhail-pyrev mentions in a comment:, First number is weekday of first day of the month, second number is number of days in said month.

	Ours	Use calendar.monthrange:
		>>> from calendar import monthrange
		>>> monthrange(2011, 2)
		(1, 28)
		Just to be clear, monthrange supports leap years as well:
		>>> from calendar import monthrange
		>>> monthrange(2012, 2)
		(2, 29)
		As @mikhail-pyrev mentions in a comment:, First number is weekday of first day of the month, second number is number of days in said month.